SENSING AND PERCEPTION

The Sensing and Perception Subsystem of the NASA Research Telerobot

B. Wilcox, D.B. Gennery, B. Bon, and T. Litwin
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109

1. Abstract

A useful space telerobot for on-orbit assembly, maintenance, and repair tasks must have a sensing and perception subsystem which can provide the locations, orientations, and velocities of all relevant objects in the work environment. This function must be accomplished with sufficient speed and accuracy to permit effective grappling and manipulation. Appropriate symbolic names must be attached to each object for use by higher-level planning algorithms. Sensor data and inferences must be presented to the remote human operator in a way that is both comprehensible in ensuring safe autonomous operation and useful for direct teleoperation. Research at JPL toward these objectives is described.

2. Introduction

The JPL Robotics Laboratory has been conducting sensing and perception research since the mid 1970's, when a task was undertaken to develop a breadboard Mars rover which could navigate autonomously over unknown terrain. At that time, and continuing to the present, the principal sensor modality addressed was machine vision. This arises from the fact that it is essential, both in planetary rover and orbital tasks, to sense the environment prior to actual physical contact so that contact forces can be controlled. The available non-contact sensing techniques are limited to those based on electromagnetic radiation and those based on sound. Obviously sound is not useful in vacuum and of limited use in extremely rarified atmospheres. Electromagnetic sensing can be of an active type, emitting radiation and sensing the reflection, or passive, relying on ambient radiation. Active sensing systems can give direct information such as object range, but often consume excessive power and involve mechanical scanning devices which are potentially unreliable. Thus passive electromagnetic sensing is an attractive means of accomplishing the non-contact sensing function. The only wavelengths for which large amounts of ambient radiation exist in space are those emitted by the Sun, i.e. visible light and near IR. Sensors for these wavelengths are readily available with very good spatial and temporal resolution and accuracy in the form of solid-state video cameras. This has the further advantage that the human operator can easily comprehend the raw data from these sensors using a video display.

A useful space telerobot for on-orbit assembly, maintenance, and repair tasks must have a sensing and perception subsystem which can provide the locations, orientations, and velocities of all relevant objects in the work environment. Current goals of our research are to develop technology which will allow visual acquisition and tracking of known but unlabelled objects in space with sufficient speed and accuracy to permit effective grappling and manipulation. Examples of the potential uses of such technology are robotic systems for capturing satellites which have arbitrary and unknown motion, and robotic systems for construction in space. The vision system currently under development includes custom-designed image-processing hardware, and acquisition and tracking software running on a general purpose computer (Figure 1).

The machine vision system at JPL is designed to acquire and track polyhedral objects moving and rotating in space, using two or more cameras, programmable image-processing hardware, and a general purpose computer for high-level functions. The image-processing hardware is called PIFEX, for "Programmable Image Feature Extractor," and is capable of performing a large variety of operations on images and on image-like arrays of data. Acquisition utilizes image locations and velocities of features extracted by PIFEX to determine the 3-dimensional position, orientation, velocity and angular velocity of an object. Acquisition takes several seconds, but is adequate to initialize the object tracker. Tracking correlates edges detected in the current image with edge locations predicted from an internal model of the object and its motion, continually updating velocity information to predict where edges should appear in future frames. Once tracking has begun, it processes some 10 frames per second, thus allowing real-time tracking of objects.

3. PIFEX

PIFEX is a pipelined-image processor being built in the JPL Robotics Lab. It is a programmable system that will perform elaborate computations whose exact nature is not fixed in the hardware, and that can handle multiple images. It thus is more versatile than previous pipelined-image processors. It also is a very powerful system. A moderate-sized PIFEX costing less than $100,000 will be able to perform about 10^{10} 12-bit elementary operations per second. PIFEX is a powerful, flexible tool for image processing and low-level computer vision. It also has applications in other two-dimensional problems such as route planning for obstacle avoidance and the numerical solution of two-dimensional partial differential equations.

PIFEX contains three types of programmable operators (Figure 2): convolvers, neighborhood comparison operators, and binary functions. The convolvers use a 3-by-3 kernel. Larger kernels can be simulated through the use of multiple convolvers, although this is efficient only in special cases. The neighborhood comparison operators produce a nonlinear function of the pixels in a 3-by-3 neighborhood. They are useful for such things as finding peaks, ridges, valleys, and zero crossings, as well as for region growing, shrinking, and other cellular operations. The binary functions receive two inputs and compute any desired function of their corresponding pixel values, by means of table lookup with linear interpolation. PIFEX consists of an array of identical modules, each of which contains two convolvers, one binary function, and one neighborhood comparison operator.

The modules are connected in a regular pattern in which each of two outputs from each module branches to the inputs of several

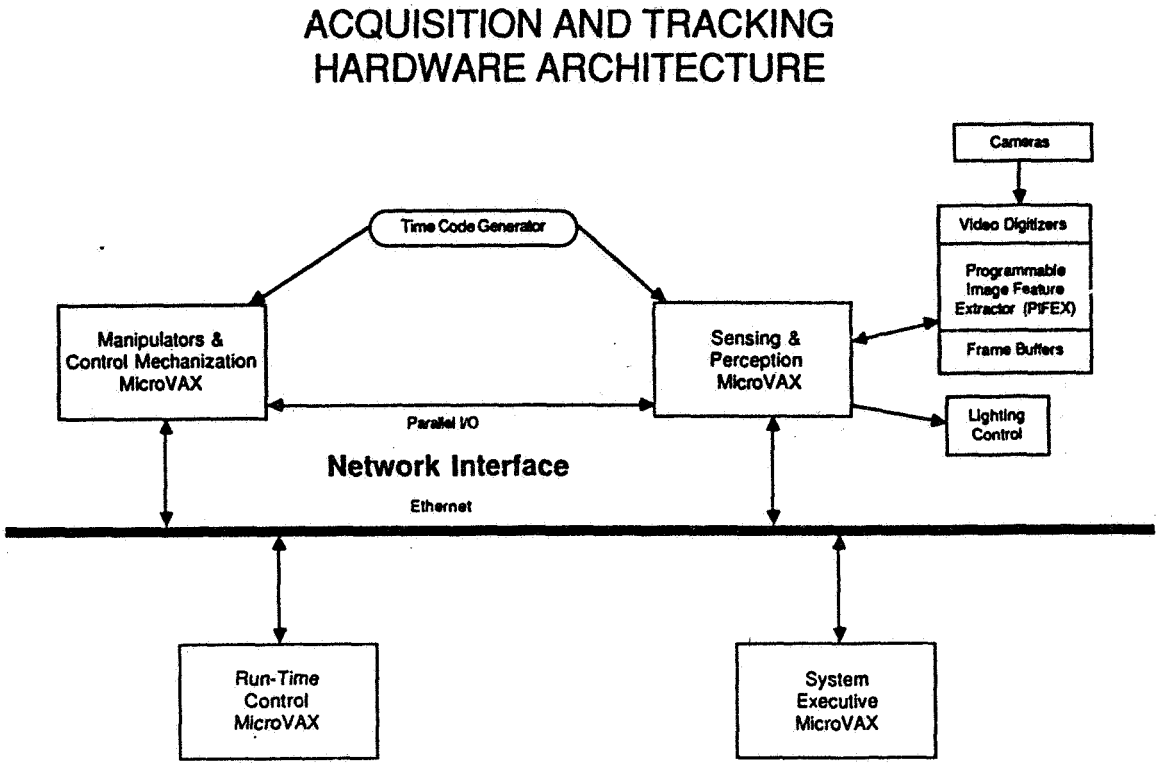

Figure 1. Acquisition and Tracking Hardware Architecture

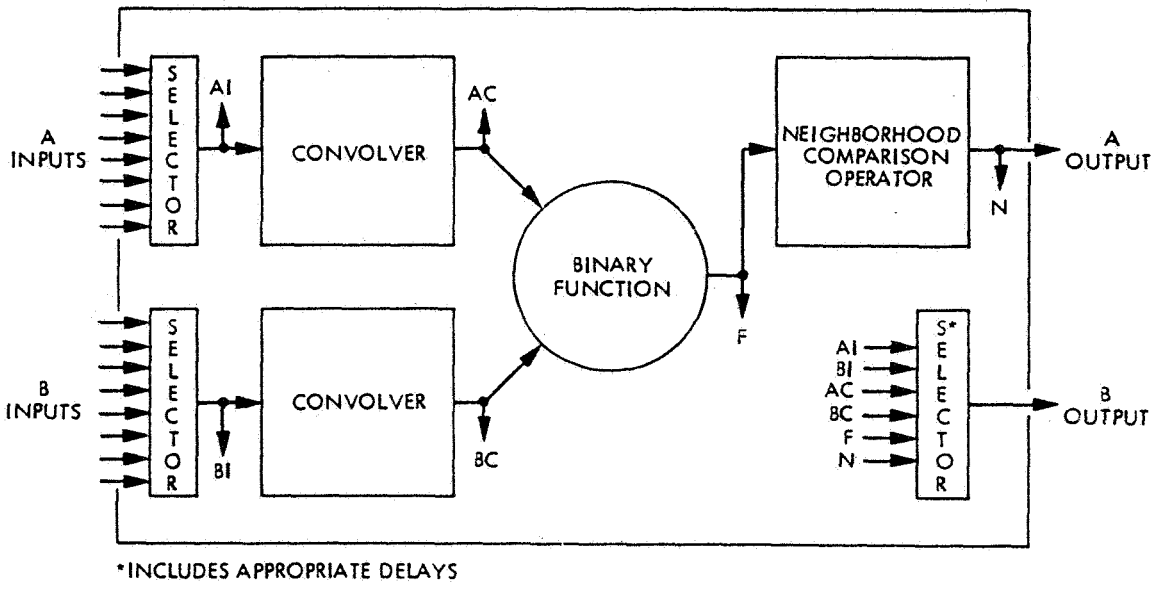

Figure 2. PIFEX module

different modules. The outputs from the modules in each column in the pattern are connected to the inputs of modules in the next column, so that the main data flow is considered to be from left to right. In this way, synchronism is achieved, since all of the modules in a given column (except for the wrap-around of rows discussed below) are processing corresponding pixels at the same time. Different rows of modules correspond to parallel data paths, but these different paths can communicate with each other because of the branching of the connections from one column to the next.

Each row is considered to wrap around to form a loop. The fanout pattern continues cyclically around these loops, except that after one particular column there are switches that can break each connection between the output of a module and the fanout to the next column, so that outputs can be extracted here and inputs can be inserted. This row wrap-around feature is important for efficient coding of algorithms that vary greatly in the width and length of data paths that they require, since an algorithm that requires a long path can wrap around several times, using only as many parallel data paths at any point as it needs. The upward bias of the fanout helps in spiraling the paths upwards in order to avoid collisions. (Since the pixels have been delayed by different amounts on different times around, ordinarily data from these different paths should not be combined with each other.) Also, each column wraps around to form a loop, and the fanout pattern is cycled invariantly around the loops. This feature is convenient for algorithms that just barely fit, since crossing the boundary that otherwise would exist at the top and bottom may help in making the necessary connections. In particular, the column wrap-around feature makes it practical to extract the outputs at the same rows at which the inputs are inserted in cases where row wrap-around is used, so that modules are not wasted.

The two wrap-around features combined cause the the interconnections of the modules in PIFEX to have the topology of a torus. There is a cut around the torus at one place to allow inputs (from image buffers or TV cameras) and outputs (to image buffers) to be switched in (as stated above), under control of the host computer.

It is planned that the initial version of PIFEX will have 5 columns and about 24 rows. (Thus it would be possible to code algorithms that vary from requiring a data path 24 modules wide and 5 modules long to requiring a data path one module wide and 120 modules long, without having to use separate passes through PIFEX on separate frame times.)

Even though the chosen approach results in a physically larger device (and perhaps greater cost if produced in quantity) than other possible approaches, it has the advantages of quicker and less expensive development (because of the need for fewer types of complicated custom VLSI chips), ease of computing arbitrary functions (because of the generality of the table-lookup functions), and easy growth to a more powerful system (because of the modular concept with the regular interconnection pattern).

PIFEX has been described in more detail elsewhere.[1,2]

4. Acquisition and Tracking

The organization of the acquisition and tracking system is shown in Figure 3. Its operation will be described briefly in this section.

The Feature Tracker detects features in the images from each camera, tracks them as they move over time, smooths their two-dimensional positions, and differentiates the positions to obtain their two-dimensional velocities in the image plane. (The features currently used correspond to the vertices of a polyhedral object.) Features that are not moving, are moving too fast, or do not remain sufficiently long are rejected. Future versions of the Feature Tracker may also measure other properties of the features in addition to position, such as orientation, to aid in stereo matching and in matching to the object model. The Feature Tracker will run primarily in PIFEX when it becomes available.

When enough features are being tracked, the Motion Stereo module uses the information from all of the cameras for some particular time to compute the partial three-dimensional information. For a single camera, the object range is completely indeterminate, but the relative ranges of the features are determined using the assumption that they are connected to a rigid, moving object. For multiple cameras, the absolute ranges of the features in general can be determined. This includes the three-dimensional position of each feature (from any camera), an estimate of its position accuracy as given by a 3-by-3 covariance matrix, and estimates of the velocity and angular velocity of the object. All of this information is based on nominal values of unity for scale factor and zero for bias. In addition, a 2-by-2 covariance matrix of the uncertainty in these nominal values of scale factor and bias is estimated. The motion stereo algorithm has been described in more detail elsewhere.[3]

The Stereo Matcher refines this information and computes estimates of the scale factor and bias. It uses a general matching process based on a probabilistic search.[4] In this process, features from one camera are matched one at a time to features from another camera in order to build a search tree. For each combination of trial matches, a least-squares adjustment is done for the scale factor and bias that produces the best agreement of the matched features. The discrepancies in the adjusted positions of the matched features compared to their accuracies are used to compute a probability for each match combination, and these probabilities are used to prune the search. If there are more than two cameras, the current plan is for the Stereo Matcher to use only a specified pair for matching, but more elaborate arrangements are possible.

The Model Matcher matches the three-dimensional feature positions (and any other feature information available) to those of the object model in order to determine the three-dimensional position and orientation of the object, by using a search process similar to that in the Stereo Matcher. This information is valid for the time at which the features had these positions. However, time has elapsed since then, while the computations in the Motion Stereo module, Stereo Matcher, and Model Matcher were being done.

Meanwhile, the Feature Tracker, running concurrently with the other modules, still has been tracking the features (those that have remained visible). The latest positions of these features, together with the information from the model matcher that indicates which object features they match, are used by the Tracking Initializer to update the object position and orientation to the time of this most recent data. Also, the two-dimensional velocities from the Feature Tracker are used to compute the three-dimensional velocity and angular velocity of the object for this time. The solution for position and orientation in the Tracking Initializer is an iterative nonlinear weighted least-squares adjustment. The initial approximation for this iterative solution is obtained from the old position and orientation from the Model Matcher, extrapolated to the new time by using the velocity and angular velocity from the Motion Stereo module, as corrected by using the scale factor and bias estimates from the Stereo Matcher. The accuracy estimates of the solution, in the form of a 12-by-12 covariance matrix of position, orientation, velocity, and angular velocity, also are produced.

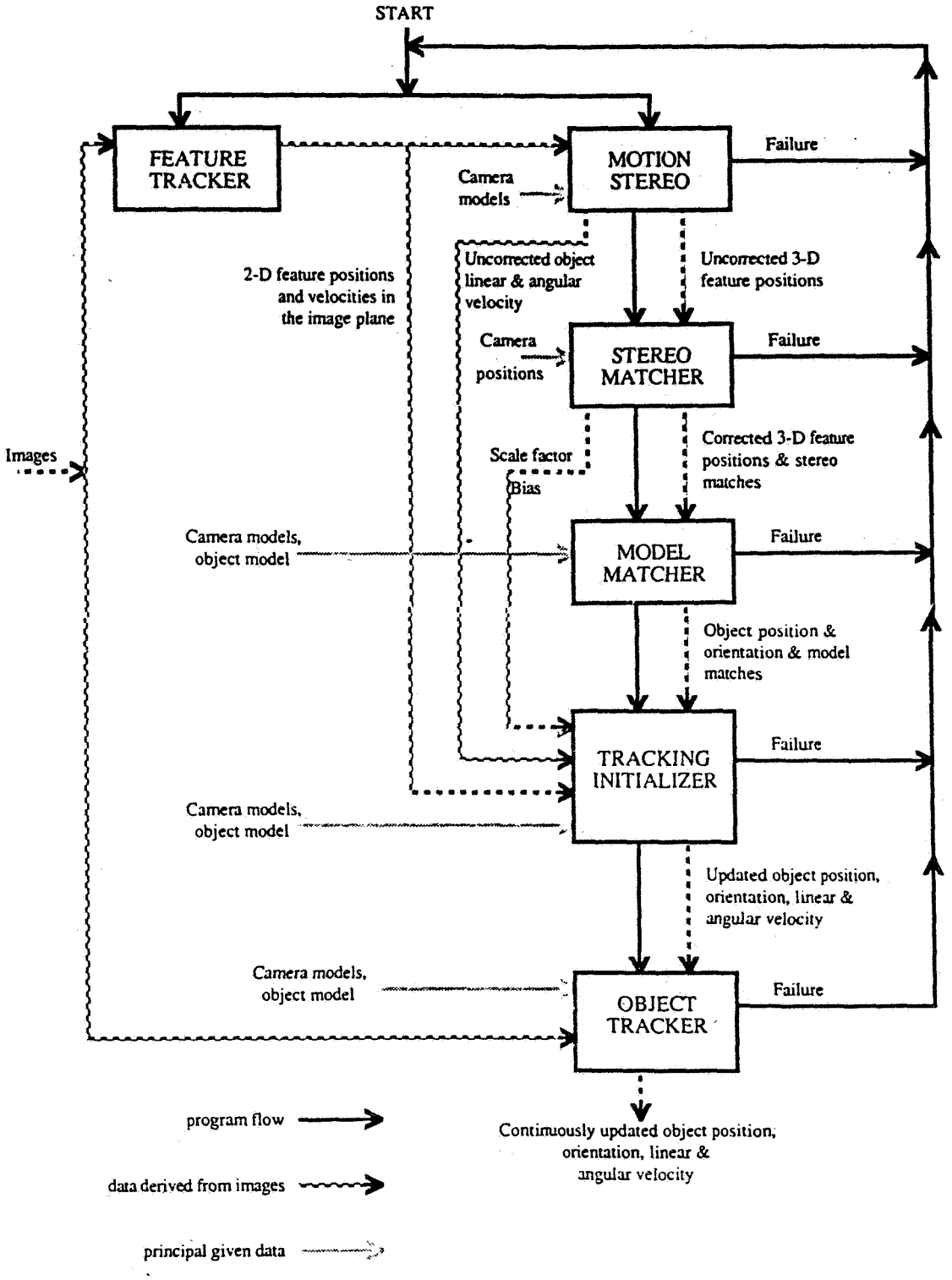

Figure 3. Acquisition and Tracking Software Architecture.

The position, orientation, velocity, angular velocity, and their covariance matrix from the Tracking Initializer are used as initial conditions in the Object Tracker. It rapidly and accurately updates this information. Currently, the features that it looks for in the images are the object edges. Using edges produces more complete information than using vertices. Edges can be used easily here, because the one-dimensional information from edge elements suffices once the approximate object position and orientation are known. The edges currently are detected by IMFEX,[5] which is a nonprogrammable precursor to PIFEX and computes an approximation to the Sobel operator. When PIFEX is available, it will detect the edges and perform a portion of the computation involving them.

The object tracker works in a loop with the following major steps: prediction of the object position and orientation for the time at which a picture is taken by extrapolating from the previously adjusted data (or from acquisition data when starting); detection of features by projecting into the picture to find the actual features and to measure their image positions relative to the predictions; and the use of the resulting data to adjust the position, orientation, and their time derivatives so that the best estimates for the time of the picture are obtained. The object tracker has been described elsewhere.[6]

It is possible for any of these modules to fail because of poor data. A failure in any of them causes the acquisition process to start over. As new features become visible, they may contain good enough information for successful acquisition and tracking. (The Object Tracker can track through regions of data too poor to allow acquisition to occur. If it fails, the re-acquisition probably will not succeed immediately, but eventually the object may move into a region of sufficient visibility for acquisition.)

Notice that in the Model Matcher, the Tracking Initializer, and the Object Tracker stereo information is used implicitly. That is, stereo matching between cameras need not be done for all features used by the Model Matcher and the Tracking Initializer, and it is not done for any features in the Object Tracker. Instead, features are matched directly to the object model. (In all three modules, these features can come separately from each camera, but in the Model Matcher and Tracking Initializer, those features that have been matched between cameras by the Stereo Matcher are used as units.) This process produces accurate stereo depth information even if the same features are not seen by different cameras, because of the rigid-body constraint in the object model.

This approach can be extended directly to multiple object recognition. Since one of the outputs of the model matcher is a probability of correct match, several of these matchers working in parallel with different object models could perform the recognition function. However, if a large number of different objects need to be recognized, additional modules would need to be created to classify the feature patterns into one of several groups before an attempt to make a detailed match. These broad classifications of objects might be made on the basis of the presence of cylindrical or spherical surfaces or the number of features of a given type (edge, vertex, etc.).

5. Camera calibration

The grappling of a spinning or tumbling satellite requires that the manipulator control system and the machine vision system agree on the 3-D positions of objects in the work volume. To achieve this correspondence, a calibration fixture has been fabricated that is used for both manipulator calibration[7] and camera calibration. This fixture has an array of dots machined on a black-anodized aluminium plate, mounted on a framework which can be affixed to the floor of the research facility in any one of nine pre-measured positions. These positions include three different planes for the face of the plate, so that the dots on the array are seen by the cameras at three different distances, allowing accurate determination of the camera parameters.

The first step in camera calibration is to capture images from the various cameras of the calibration fixture in each of the measured positions. Manual input consists of the following: the camera number, the position number of the measured fixture position, the 3-D coordinates of these positions, the spacing of the dots, the diameter of the dots, the number of rows and columns of dots, the nominal focal length of the camera, the nominal pixel spacing, and the approximate 3-D position of the camera. At present, the operator designates the corner dots. The result is a set of points, with for each point its 3-D position and its measured 2-D position.

First, the approximate dot spacing a (in pixels) in the image plane is computed from the designated corner dot positions. Then the approximate Gaussian function for filtering is defined so that its standard deviation is half of the average dot spacing. The image is low-pass filtered by convolving with a one-dimensional Gaussian function first along the columns and then along the rows, and the result is subtracted from the original image to obtain the high-pass-filtered image.

A histogram of the high-pass-filtered image b_h is computed for the portion of the image which is expected to include all but the outer rows and columns of dots, with buckets for every integer from -255 to 255. This is summed and normalized to produce the cumulative distribution c_h. The predicted portion of area covered by the dots is computed from the known size and spacing of the dots. Then values halfway between this and 0 and 1 are computed, and the brightness values for which the cumulative histogram is equal to these values are found. The average of these two brightness values is used as the threshold.

Every pixel of the high-pass-filtered image within the expected area of the dots whose value exceeds that of the threshold is tentatively assumed to be part of a dot. Every connected (by four-neighbor connection) area of such pixels is examined to see if it forms a good dot. Its area should be within four pixels plus 10% of the expected value, and the Euclidean distance of its border pixels from the centroid of its pixel positions should not vary by more than one pixel plus 5%. The dots that pass these tests are used, and the others are rejected. For each dot that passes, the centroid of its pixel positions is used as the 2-D dot position (to sub-pixel accuracy). The 3-D dot position is obtained from the known dot spacings, with the individual dots being identified by progressing one dot at a time from a known corner dot according to the expected dot image spacing.

6. Camera Model Adjustment

The actual camera model adjustment is performed by a least-squares adjustment, which finds the set of camera model parameters that minimizes the sum of the squares of the differences btween the predicted postions and measured postitions (in two dimensions) of the dots on the calibration fixture. The form of the camera model is that described in Yakimovsky and Cummingham 1978[8], although we will probably add two terms for lens distortion to the model later. The least-squares adjustment is performed iteratively, since the problem is nonlinear. Also, in case the dot finder makes mistakes, automatic editing is done to remove bad dots, using the method described in Gennery 1980 and Gennery 1986.[9,3]

7. Status

A wire-wrapped prototype PIFEX module has been produced and debugged, using a version of the convolver composed of three custom VLSI chips (plus the line buffers). A printed circuit layout is being designed for use with a single-chip convolver, leading to production of a PIFEX with about 120 modules. A high-level language for programming PIFEX has been designed, and a compiler will be written for it.

The acquisition and tracking system has been designed, and most of it has been coded in Pascal for the microVAX-II. The Feature Tracker, Motion Stereo module and Stereo Matcher have executed successfully. The Model Matcher is still under development, and coding has begun on the Tracking Initializer. The Object Tracker was running on a different computer from the VAX presently in use; it has been translated for use on the VAX but has yet to run on real images there. Once all modules are working, optimization and integration will begin. Finally, when a sufficiently large PIFEX is available, appropriate parts of acquisition and tracking, including much of the Feature Tracker, will be programmed into PIFEX, thus increasing the speed and robustness of the system.

8. Acknowledgments

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under contract with the National Aeronautics and Space Administration.

9. References

[1] D. B. Gennery and B. Wilcox, "PIFEX: An Advanced Programmable Pipelined-Image Processor," JPL Publication 84-97, Jet Propulsion Laboratory, Pasadena, CA, 1984.

[2] D. B. Gennery and B. Wilcox, "A Pipelined Processor for Low-Level Vision," *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, June 1985, pp. 608-613.

[3] D. B. Gennery, "Stereo Vision for the Acquisition and Tracking of Moving Three-Dimensional Objects," in *Techniques for 3-D Machine Perception* (A. Rosenfeld, ed.), North-Holland, 1986.

[4] D. B. Gennery, "A Feature-Based Scene Matcher," *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver, August 1981, pp. 667-673.

[5] R. Eskenazi and J. M. Wilf, "Low-Level Processing for Real-time Image Analysis," JPL Publication 79-79, Jet Propulsion Laboratory, Pasadena, CA, 1979.

[6] D. B. Gennery, "Tracking Known Three-Dimensional Objects," *Proceedings of the AAAI Second National Conference on Artificial Intelligence*, Pittsburgh PA, August 1982, pp. 13-17.

[7] S. Hayati and M. Mirmirani, "Improving the absolute positioning accuracy of robot manipulators", *Journal of robotic systems*, Vol II, no 4, 1985.

[8] Y. Yakimovsky and R. T. Cunningham, "A System for Extracting Three-Dimensional Measurements from a Stereo Pair of TV Cameras," *Computer Graphics and Image Processing* 7, pp. 195-210 (1978).

[9] D. B. Gennery, "Modelling the Environment of an Exploring Vehicle by Means of Stereo Vision," AIM-339 (STAN-CS-80-805), Stanford University, Computer Science Dept, June 1980.